Handling logs

WHAT'S A LOGGER?

- A logger is a class that we use to show messages in console

For example, this:

```
Gdx.app.log("TAG", "Informative message");
```

Would look like this:

```
TAG: Informative message
```

Why Shouldn't I Use System.out.println() ("SOUT") Instead?

- SOUT only works for desktop
- Your game can reach production with log messages
- With a logger you can control which kind of messages you show

HOW DO I USE IT?

We Have 3 Different Methods:

```
Gdx.app.debug("TAG", "Message");
Gdx.app.log("TAG", "Message");
Gdx.app.error("TAG", "Message", new ExampleException());
```

DEBUG:

- Used to show messages only while developing our game
- We should disable it when publishing the game

LOG:

- Used to show informative messages to the user
- Disabling it also disables debug

ERROR:

- Used to show runtime error messages and exceptions to the user
- Disabling it also disables debug and log
- You can attach an exception to it

EXAMPLE USE CASES

DEBUG:

```
this.y = MathUtils.random(HEIGHT_OFFSET);
Gdx.app.debug(“FLOWER”, “Y is now “ + y);
```

LOG:

```
if (checkForCollision()) {
    restart();
    Gdx.app.log("INFO", "Game restarted");
}
```

ERROR:

```
String fileName = "file.txt";
try {
    FileInputStream in = new FileInputStream(fileName);
} catch (FileNotFoundException e) {
    Gdx.app.error("DATA", "Couldn't open file " + fileName, e);
}
```

CHANGING THE LOG LEVEL

By default the log level is set to Info, but we can change using this method:

```
Gdx.app.setLogLevel(logLevel);
```

LogLevel Can Be:

- **Application.LOG_NONE**: mutes all logging.
- **Application.LOG_DEBUG**: shows all messages.
- **Application.LOG_INFO**: shows error and log messages.
- **Application.LOG_ERROR**: shows only error messages.

DEBUG → includes → **INFO** → includes → **ERROR**